



第二讲：分类回归树

—— 暂别XGBoost

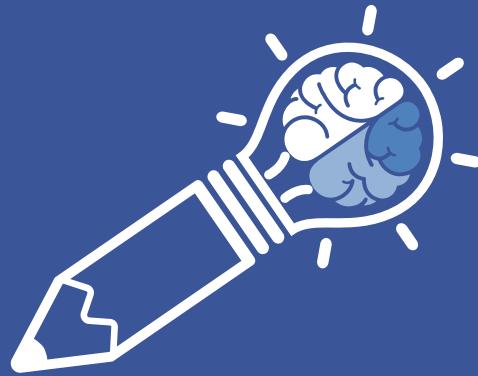
AI100学院
2017年6月



► Roadmap

- 监督学习
- 分类回归树
- 随机森林





一、监督学习



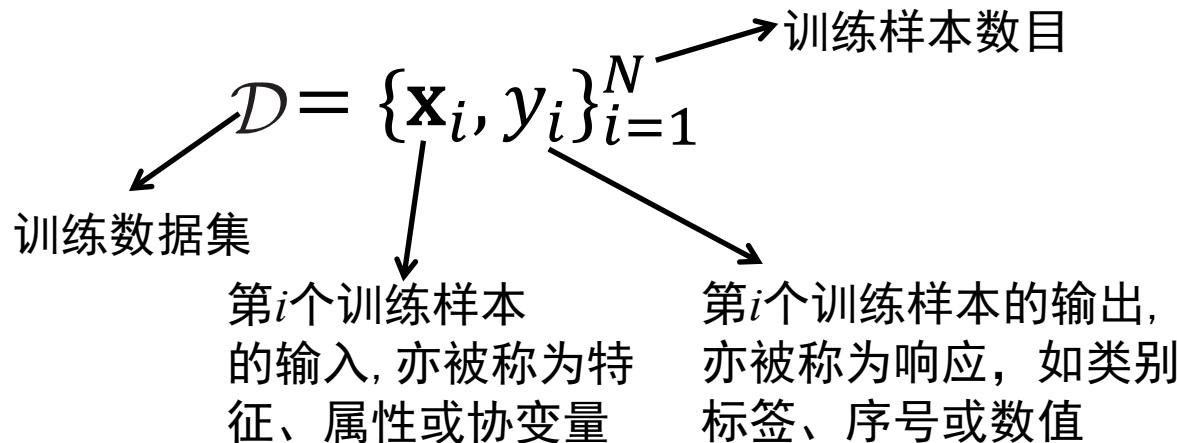
► 监督学习

- 模型
- 参数
- 目标函数
 - 损失函数
 - 正则项
- 优化



► 监督学习

- **监督学习**：给定带标签的 N 个训练样本 $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ ，学习到一个 $\mathbf{x} \rightarrow y$ 的映射 f ，从而对新输入的 \mathbf{x} 进行预测



• 监督学习：给定带标签的训练样本 $= \{\mathbf{x}_i, y_i\}_{i=1}^N$ ，学
习一个 $\mathbf{x} \rightarrow y$ 的映射 / 从而对新输入的 \mathbf{x} 进行预测

► 回归 / 分类

- 监督学习：给定带标签的训练样本 $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ ，学习到一个 $\mathbf{x} \rightarrow y$ 的映射 f ，从而对新输入的 \mathbf{x} 进行预测
- 若标签 y 为离散值 / 类别，称为分类
 - 如根据蘑菇的颜色、形状等判断蘑菇是否有毒： $y=1$, 可食用; $y=0$, 有毒
- 若标签 y 为连续值，称为回归
 - 如根据房屋的位置、面积等估计房屋价格， y 为房屋价格

► 模型—回归

- 监督学习：给定带标签的训练样本 $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ ，学习到一个 $\mathbf{x} \rightarrow y$ 的映射 f ，从而对新输入的 \mathbf{x} 进行预测
- 模型：对给定的输入 \mathbf{x} ，如何预测其标签 \hat{y}
- 最简单的模型：线性模型 $\hat{y} = f(\mathbf{x}) = \sum_j w_j x_j = \mathbf{w}^T \mathbf{x}$
- 对回归问题： $\hat{y} = f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$
 - 称为线性回归

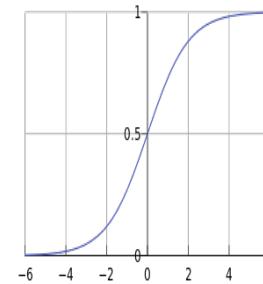
► 模型—分类

- 对两类分类问题，标签为二值变量， $p(y|x)$ 为Bernoulli分布
 - $y \in [0,1], p(y|x, w) = \text{Ber}(y; f(x))$
 - 其中 $f(x) = p(y=1|x)$
 - 线性模型：计算输入x的线性组合： $\sum_j w_j x_j = w^T x$
 - 但由于 $f(x) \in [0,1]$ ，但经过一个sigmoid函数做变换（S形函数）：
$$\text{sigm}(\eta) = \frac{1}{1 + \exp(-\eta)} = \frac{\exp(\eta)}{\exp(\eta) + 1}$$
 - 亦被称为logistic函数或logit函数



综合起来： $p(y|x, w) = \text{Ber}(y | \text{sigm}(w^T x))$

- 和线性回归相似，因此被称为**Logistic回归**（虽然是分类）



► 模型—分类 (cont.)

- 为什么用logistic函数？
- 来自神经科学：
 - 神经元对其输入加权和： $\mathbf{w}^T \mathbf{x}$
 - 如果该和大于某阈值 $\mathbf{w}^T \mathbf{x} > t$ ， 神经元发放脉冲
- Logistic回归：当 $p(y=1 | \mathbf{x}, \mathbf{w}) > p(y=0 | \mathbf{x}, \mathbf{w})$ 时发放
- 定义Log Odds Ratio: $LOR(\mathbf{x}) = \log \frac{p(y=1 | \mathbf{x}, \mathbf{w})}{p(y=0 | \mathbf{x}, \mathbf{w})}$

$$= \log \left[\frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})} \frac{1 + \exp(-\mathbf{w}^T \mathbf{x})}{\exp(-\mathbf{w}^T \mathbf{x})} \right]$$

$$= \log \left[\exp(\mathbf{w}^T \mathbf{x}) \right] = \mathbf{w}^T \mathbf{x}$$



因此 iff $LOR(\mathbf{x}) = \mathbf{w}^T \mathbf{x} > 0$ 发放。



► 模型参数

- 确定模型类别后，模型训练转化为求解模型参数
- 如对线性模型 $\hat{y} = f(\mathbf{x}) = \sum_j w_j x_j = \mathbf{w}^T \mathbf{x}$,
- 参数为 $\theta = \{w_j | j = 1, \dots, D\}$
 - D 为特征维数
- 求解模型参数：目标函数最小化

▶ 目标函数

- 目标函数通常包含两项：损失函数和正则项

$$J(\theta) = \sum_{i=1}^N L(y_i, f(\mathbf{x}_i; \theta)) + \Omega(\theta)$$

参数 θ 对应的损失函数
度量参数对应的模型与
训练数据的拟合程度

参数 θ 对应的正则项
对模型的复杂度施加惩罚

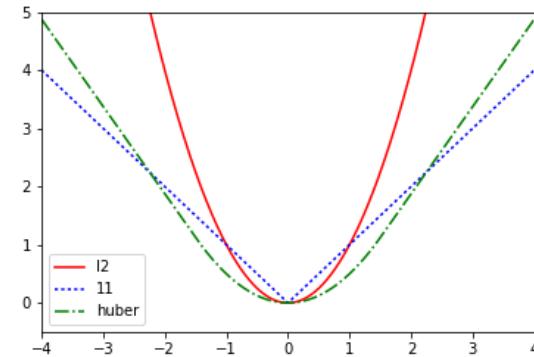
► 损失函数—回归

- 损失函数：度量模型预测值与真值之间的差异
- 对回归问题：令残差 $r = y - f(\mathbf{x})$
 - L2损失： $L(r) = (r)^2$

- L1损失： $L(r) = |r|$

对噪声
不敏感

- Huber损失：
$$L_\delta(r) = \begin{cases} (r)^2/2 & if |r| \leq \delta \\ \delta|r| - (r)^2/2 & if |r| > \delta \end{cases}$$



► 损失函数——分类

- 损失函数：度量模型预测值与真值之间的差异
- 对分类问题

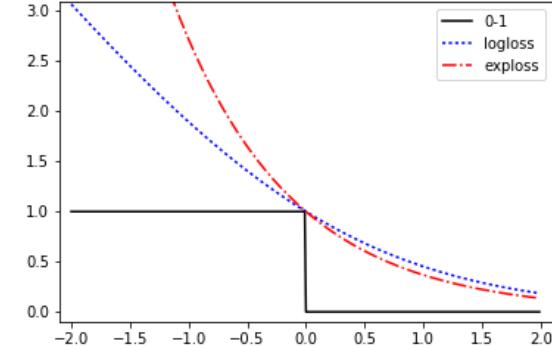
– 0-1损失: $L(y, f(\mathbf{x}; \boldsymbol{\theta})) = \begin{cases} 0 & \text{if } y = f(\mathbf{x}; \boldsymbol{\theta}) \\ 1 & \text{if } y \neq f(\mathbf{x}; \boldsymbol{\theta}) \end{cases}$

– Logistic损失: 亦称负log似然损失

$$- L(y, f(\mathbf{x}; \boldsymbol{\theta})) = -y \log(f(\mathbf{x}; \boldsymbol{\theta})) - (1 - y) \log(1 - f(\mathbf{x}; \boldsymbol{\theta}))$$

– 指数损失: $L(y, f(\mathbf{x}; \boldsymbol{\theta})) = \exp(-y f(\mathbf{x}; \boldsymbol{\theta}))$

- AdaBoost中采用，对噪声敏感



► Logistic损失

$$p(y; \theta) = \text{Ber}(y; \theta) = \theta^y(1 - \theta)^{1-y}$$

- Logistic回归： $p(y|\mathbf{x}, \boldsymbol{\theta}) = \text{Ber}(y; f(\mathbf{x}; \boldsymbol{\theta}))$
- Logistic损失亦被称为**负log似然损失**
- $NLL(\mathbf{w}) = -\sum_{i=1}^N \log(p(y_i|\mathbf{x}_i; \mathbf{w}))$
- $= -\sum_{i=1}^N \log(f(\mathbf{x}_i; \boldsymbol{\theta})^{y_i}(1 - f(\mathbf{x}_i; \boldsymbol{\theta}))^{1-y_i})$
- $= -\sum_{i=1}^N y_i \log(f(\mathbf{x}_i; \boldsymbol{\theta})) + (1 - y_i) \log(1 - f(\mathbf{x}_i; \boldsymbol{\theta}))$
- $= \sum_{i=1}^N L(y_i, f(\mathbf{x}_i; \boldsymbol{\theta}))$
- 其中 $L(y, f(\mathbf{x}; \boldsymbol{\theta})) = -y \log(f(\mathbf{x}; \boldsymbol{\theta})) - (1 - y) \log(1 - f(\mathbf{x}; \boldsymbol{\theta}))$



► Recall : 负log似然损失

- 似然 : $Likelihood(\theta) = \prod_{i=1}^N p(y_i | \mathbf{x}_i; \theta)$
 - 数据中的样本为独立同分布 (IID) 样本
- Log似然: $l(\theta) = \log(Likelihood(\theta)) = \sum_{i=1}^N \log(p(y_i | \mathbf{x}_i; \theta))$
 - Log: 数值计算更稳定
 - 很多分布的pdf为指数形式, log运算后更简单
- 负log似然 : $NLL(\theta) = -l(\theta) = -\sum_{i=1}^N \log(p(y_i | \mathbf{x}_i; \theta))$
 - 很多优化问题习惯求极小值
- 最小化负log似然损失等价于**最大化似然**
 - 极大似然估计 (Maximize Likelihood Estimator , MLE)



其实L2损失也可以看成负log似然损失 : $p(y|\mathbf{x}, \theta) = N(y; f(\mathbf{x}; \theta), \sigma^2)$

正态分布

<http://www.ai100.ai/>



▶ 正则项

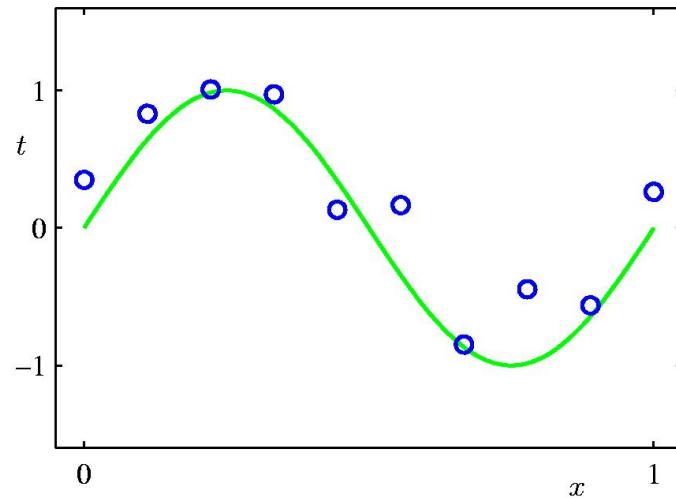
- 不能只选择损失最小的模型，因为复杂的模型可能和训练数据拟合得非常好，在训练集上可以损失几乎为0
- 但测试数据和训练数据虽然是来自同分布的独立样本，但只是分布相同，随机变量取值会变化
- 训练数据中可能会有噪声，模型不应该将噪声包含在内
- 复杂模型（预测）不稳定：方差大
- 所以需要控制模型复杂度
- 正则项：对复杂模型施加惩罚



► 例：sin曲线拟合

$$Y = \sin(2\pi X) + \varepsilon$$

$$X \sim Uniform[0,1], \quad \varepsilon \sim N(0, 0.3^2)$$

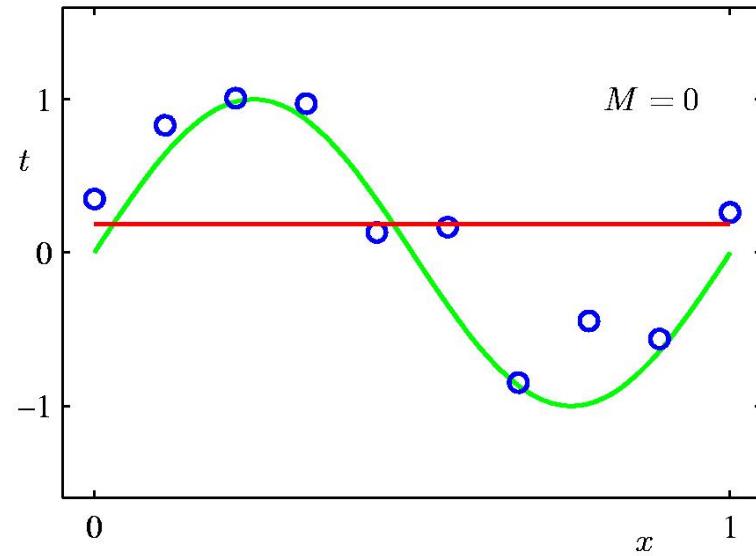


样本数 $N=10$
用 M 阶多项式拟合：

$$\hat{y} = \sum_{j=1}^M w_j x^j$$



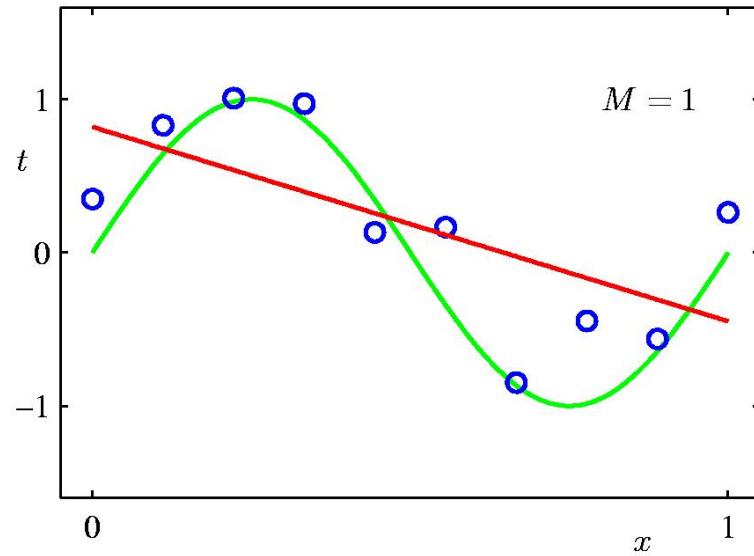
► 例：sin曲线拟合 (2)



0阶多项式拟合



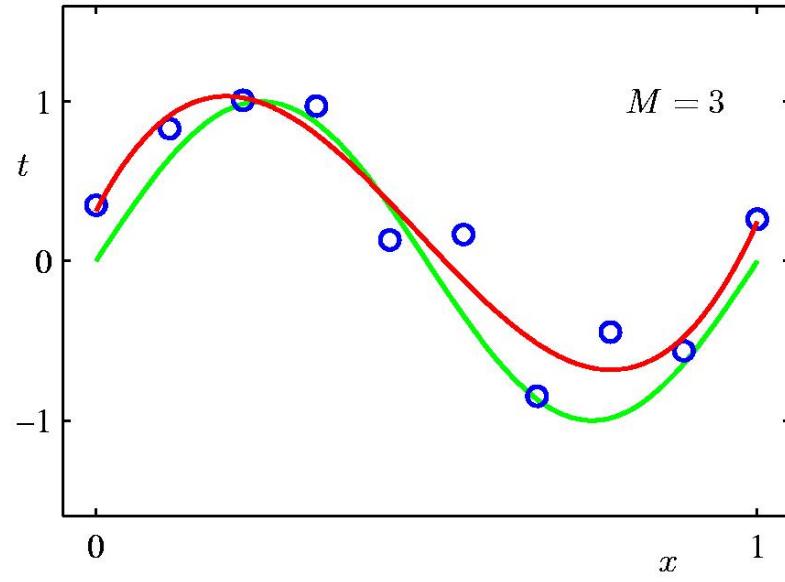
► 例：sin曲线拟合 (3)



1阶多项式拟合



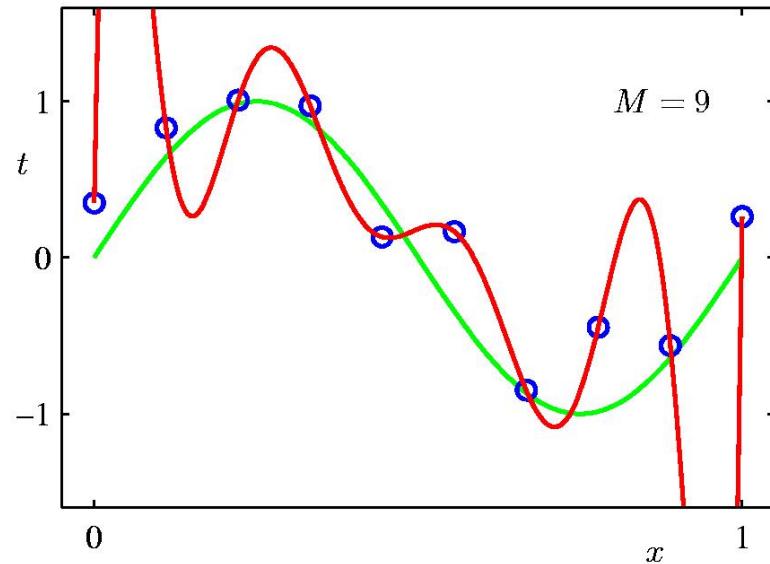
► 例：sin曲线拟合 (4)



3阶多项式拟合



► 例：sin曲线拟合 (5)



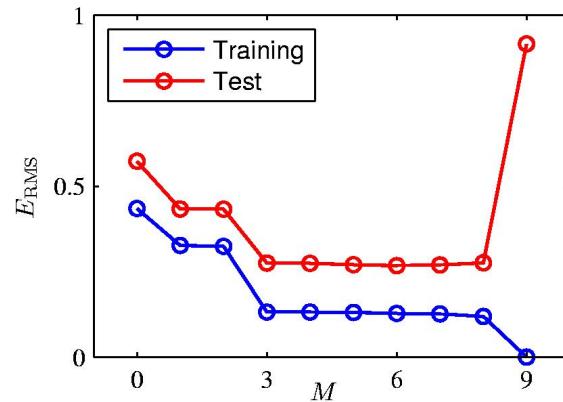
9阶多项式拟合



► 例：sin曲线拟合（6）

- 过拟合：

- 当模型复杂度增加时，训练误差继续下降，甚至趋近于0，而测试误差反而增大



$$E_{RMS} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}$$

► 训练集上的误差 \neq 测试集上的误差

- 推广性(generalization)：学习器应该在新的测试数据上表现好
- 复杂的曲线不能泛化/推广到新数据上：根据特定输入调制得太好，而不是真正建模 x 与 y 之间的关系
 - 被称为数据过拟合(overfitting)



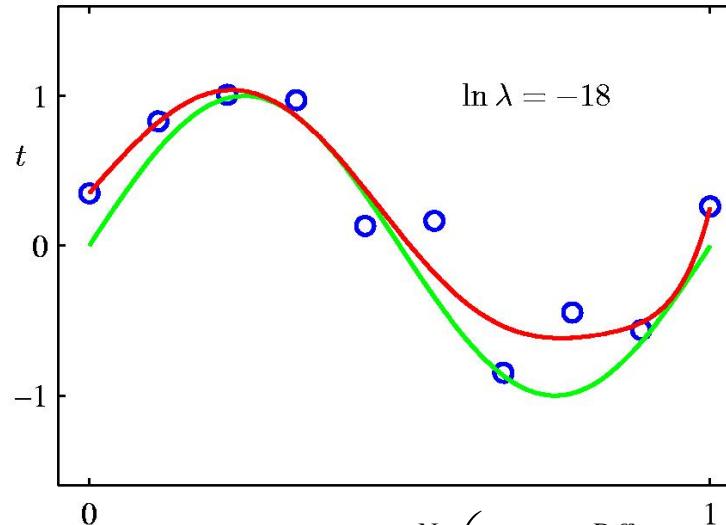
► 例：sin曲线拟合 (7)

- 回归系数：

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43

► 例：sin曲线拟合 (8)

- 增加L2正则

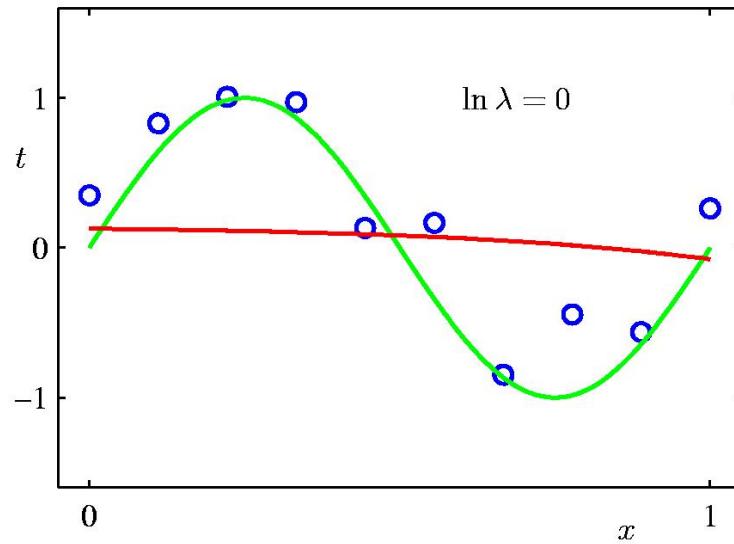


岭回归：最小化 $RSS_{ridge}(\lambda) = \sum_{i=1}^N \left(y_i - \sum_{j=0}^D x_j w_j \right)^2 + \lambda \sum_{j=1}^D w_j^2$



► 例：sin曲线拟合 (9)

- 欠拟合：模型太简单 / 对复杂性惩罚太多

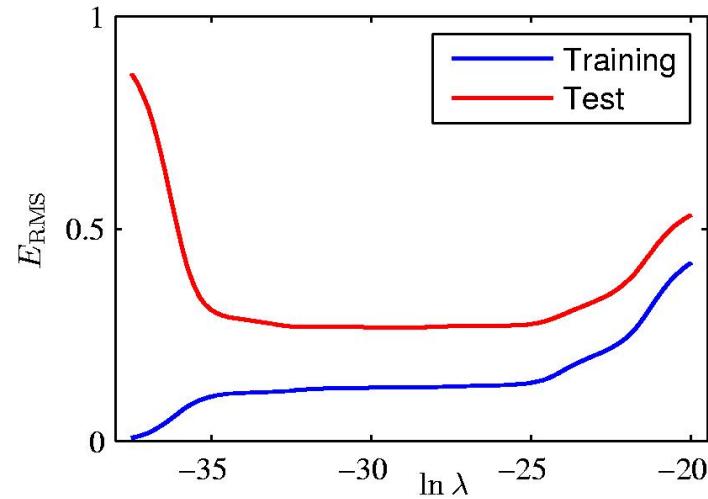


岭回归



► 例：sin曲线拟合（10）

- 不同正则参数下的训练误差vs.测试误差



岭回归



► 例：sin曲线拟合 (11)

- 岭回归系数

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
w_0^*	0.35	0.35	0.13
w_1^*	232.37	4.74	-0.05
w_2^*	-5321.83	-0.77	-0.06
w_3^*	48568.31	-31.97	-0.05
w_4^*	-231639.30	-3.89	-0.03
w_5^*	640042.26	55.28	-0.02
w_6^*	-1061800.52	41.32	-0.01
w_7^*	1042400.18	-45.95	-0.00
w_8^*	-557682.99	-91.53	0.00
w_9^*	125201.43	72.68	0.01

► 常用正则函数

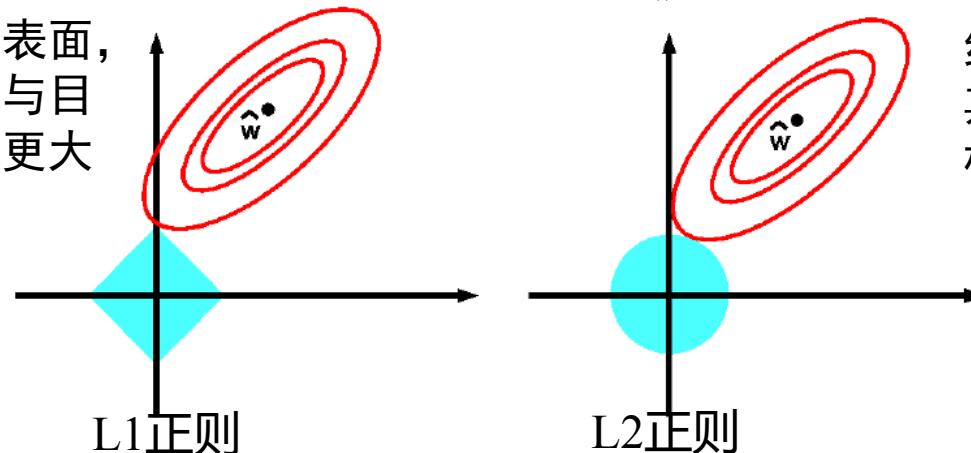
- L2正则 : $\Omega(\theta) = \lambda \|\theta\|_2^2 = \lambda \sum_{j=1}^D \theta_j^2$
- L1正则: $\Omega(\theta) = \lambda |\theta| = \lambda \sum_{j=1}^D |\theta_j|$
- L0正则 : $\Omega(\theta) = \lambda \|\theta\|_0$
 - 非0参数的数目
 - 不好优化，通常用L1正则近似

► L1正则和稀疏模型

- L1正则会使得某些参数刚好为0 → 稀疏模型
- L1正则目标函数： $J(\boldsymbol{\theta}) = \sum_{i=1}^N L(y_i, \hat{y}_i(\boldsymbol{\theta})) + \lambda \sum_{j=1}^D |\theta_j|$, 等价于带约束的优化问题： $\min J(\boldsymbol{\theta}) = \sum_{i=1}^N L(y_i, \hat{y}_i(\boldsymbol{\theta}))$, 满足约束, $\sum_{j=1}^D |\theta_j| \leq t$.

约束表面为立方体表面,
由于角更突出, 角与目
标函数相交的概率更大

角: 有些系数为0
→ 稀疏



约束表面为球形表面,
其上各点与目标函数
相交的概率相同



优化理论: 最优值为目标函数的最低水平集与约束表面相交的位置



<http://www.ai100.ai/>

► 常见线性模型的损失和正则项组合

	L2损失	L1损失	Huber损失	Logistic损失	合叶损失	ϵ -insentive损失
L2正则	岭回归			L2正则 Logistic回归	SVM	SVR
L1正则	LASSO			L1正则 Logistic回归		
L2+L1正则	Elastic net					



► 非线性模型

- 线性模型非线性化
 - 基函数： x^2 、 \log 、 \exp 、样条函数、决策树...
 - 核化：将原问题转化为对偶问题，将对偶问题中的向量点积 $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ 换成核函数 $k(\mathbf{x}_i, \mathbf{x}_j)$
 - 施加非线性变换：如（深度）神经网络中对输入的线性组合再施加非线性激活函数，然后再多层叠加

► 优化

- 简单目标函数直接求解
 - 如小数据集上的线性回归
- 更复杂问题：凸优化
 - (随机) 梯度下降
 - 牛顿法 / 拟牛顿法
 - ...

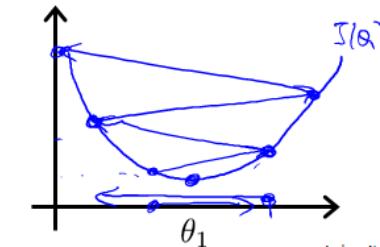


► 梯度下降 (Gradient Descent) 算法

- 梯度下降 / 最速下降算法：快速寻找函数局部极小值
 - 将函数比作一座山，我们站在某个山坡上，往四周看，从哪个方向向下走一小步，能够下降的最快。这个方向就是梯度的方向
- 梯度下降算法: 求函数 $J(\theta)$ 的最小值
 - 给定初始值 θ^0
 - 更新 θ ，使得 $J(\theta)$ 越来越小
 - $\theta^t = \theta^{t-1} - \eta \nabla_{\theta} J(\theta)$ (η : 学习率)
 - 直到收敛到 / 达到预先设定的最大迭代次数

► 梯度下降算法 (cont.)

- 下降的步伐大小 (学习率) 非常重要 : 如果太小 , 收敛速度慢 ; 如果太大 , 可能会出现 overshoot the minimum 的现象
 - 如果学习率取值后发现函数值增长了 , 则需要减小学习率的值



- 梯度下降求得的只是局部最小值
 - 二阶导数 > 0 , 则目标函数为凸函数 , 局部极小值即为全局最小值
 - 随机选择多个初始值 , 得到函数的多个局部极小值点。多个局部极小值点的最小值为函数的全局最小值。

► 随机梯度下降

- 梯度下降算法每次学习都使用整个训练集，这样对大的训练数据集合，每次学习时间过长，对大的训练集需要消耗大量的内存。此时可采用随机梯度下降(Stochastic gradient descent, SGD),每次从训练集中随机选择一部分样本进行学习。
- 更多（随机）梯度下降算法的改进版：动量(Momentum)、Nesterov accelerated gradient(NAG, 涅斯捷罗夫梯度加速)、Adagrad、 RMSprop、 Adaptive Moment Estimation(Adam)...

► L2损失函数梯度

- 若采用梯度下降求解，需要损失函数对参数 θ 的**一阶导数**
 - 二阶导数用来判断函数是否为凸函数
- XGBoost优化求解时，需要损失函数对预测函数 f 的**一阶导数**和**二阶导数**
- L2损失：
$$L(y, f(\mathbf{x}; \boldsymbol{\theta})) = \frac{1}{2}(f(\mathbf{x}; \boldsymbol{\theta}) - y)^2$$
 - $\nabla_f L(\boldsymbol{\theta}) = (f(\mathbf{x}; \boldsymbol{\theta}) - y)$
 - $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \nabla_f L(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta}),$
 - $\nabla_f^2 L(\boldsymbol{\theta}) = 1$
 - $\nabla_{\boldsymbol{\theta}}^2 L(\boldsymbol{\theta}) = (f(\mathbf{x}; \boldsymbol{\theta}) - y) \nabla_{\boldsymbol{\theta}}^2 f(\mathbf{x}; \boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta}) (\nabla_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta}))^T$

► L2损失函数梯度 (cont.)

- L2损失 : $L(y, f(\mathbf{x}; \boldsymbol{\theta})) = \frac{1}{2}(f(\mathbf{x}; \boldsymbol{\theta}) - y)^2$
 - $\nabla_f L(\boldsymbol{\theta}) = (f(\mathbf{x}; \boldsymbol{\theta}) - y)$
 - $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \nabla_f L(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta})$
 - $\nabla_f^2 L(\boldsymbol{\theta}) = 1$
 - $\nabla_{\boldsymbol{\theta}}^2 L(\boldsymbol{\theta}) = (f(\mathbf{x}; \boldsymbol{\theta}) - y) \nabla_{\boldsymbol{\theta}}^2 f(\mathbf{x}; \boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta}) (\nabla_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta}))^T$
- 对线性回归 , $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$, $\nabla_{\mathbf{w}} f(\mathbf{x}; \mathbf{w}) = \mathbf{x}$,
- $$\nabla_{\mathbf{w}} L(\mathbf{w}) = \underbrace{(f(\mathbf{x}; \mathbf{w}) - y)}_{r: \text{预测残差}} \mathbf{x}$$
$$\nabla_{\mathbf{w}}^2 L(\mathbf{w}) = \mathbf{x} \mathbf{x}^T$$



► Logistic损失梯度

- 损失函数 : $L(y, f(\mathbf{x}; \boldsymbol{\theta})) = -y \log(f(\mathbf{x}; \boldsymbol{\theta})) - (1 - y) \log(1 - f(\mathbf{x}; \boldsymbol{\theta}))$
 - $\nabla_f L(\boldsymbol{\theta}) = -\frac{y}{f(\mathbf{x}; \boldsymbol{\theta})} + \frac{1-y}{1-f(\mathbf{x}; \boldsymbol{\theta})}$
 - $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \left(-\frac{y}{f(\mathbf{x}; \boldsymbol{\theta})} + \frac{1-y}{1-f(\mathbf{x}; \boldsymbol{\theta})}\right) \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}),$
 - $\nabla_f^2 L(\boldsymbol{\theta}) = \frac{-y}{[f(\mathbf{x}; \boldsymbol{\theta})]^2} + \frac{1-y}{[1-f(\mathbf{x}; \boldsymbol{\theta})]^2}$
 - $\nabla_{\boldsymbol{\theta}}^2 L(\boldsymbol{\theta}) = \left(-\frac{y}{f(\mathbf{x}; \boldsymbol{\theta})} + \frac{1-y}{1-f(\mathbf{x}; \boldsymbol{\theta})}\right) \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}) + \left(\frac{-y}{[f(\mathbf{x}; \boldsymbol{\theta})]^2} + \frac{1-y}{[1-f(\mathbf{x}; \boldsymbol{\theta})]^2}\right) \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})(\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}))^T$

► Logistic损失函数梯度 (cont.)

- 对Logistic回归 : $f(\mathbf{x}; \mathbf{w}) = \text{sigm}(\mathbf{w}^T \mathbf{x}) = \frac{\exp(\mathbf{w}^T \mathbf{x})}{1 + \exp(\mathbf{w}^T \mathbf{x})}$
- $1 - f(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + \exp(\mathbf{w}^T \mathbf{x})}$
- $L(y, f(\mathbf{x}; \mathbf{w})) = -y \log(f(\mathbf{x}; \mathbf{w})) - (1 - y) \log(1 - f(\mathbf{x}; \mathbf{w}))$
 - $\nabla_f L(\mathbf{w}) = -\frac{y}{f(\mathbf{x}; \mathbf{w})} + \frac{1-y}{1-f(\mathbf{x}; \mathbf{w})}$
 - $\nabla_{\mathbf{w}} L(\mathbf{w}) = \underbrace{(f(\mathbf{x}; \mathbf{w}) - y) \mathbf{x}}_{r: \text{预测残差}}$
 - $\nabla_f^2 L(\mathbf{w}) = \frac{-y}{[f(\mathbf{x}; \mathbf{w})]^2} + \frac{1-y}{[1-f(\mathbf{x}; \mathbf{w})]^2}$
 - $\nabla_{\mathbf{w}}^2 L(\mathbf{w}) = f(\mathbf{x}; \mathbf{w})(1 - f(\mathbf{x}; \mathbf{w})) \mathbf{x} \mathbf{x}^T$



$$L(y, f(\mathbf{x}; \mathbf{w})) = -y \log(f(\mathbf{x}; \mathbf{w})) - (1 - y) \log(1 - f(\mathbf{x}; \mathbf{w}))$$

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = -y \times \frac{1}{f(\mathbf{x}; \mathbf{w})} \frac{\partial}{\partial \mathbf{w}} f(\mathbf{x}; \mathbf{w}) + (1 - y) \times \frac{1}{1 - f(\mathbf{x}; \mathbf{w})} \frac{\partial}{\partial \mathbf{w}} f(\mathbf{x}; \mathbf{w})$$

$$= \left[-y \times \frac{1}{f(\mathbf{x}; \mathbf{w})} + (1 - y) \times \frac{1}{1 - f(\mathbf{x}; \mathbf{w})} \right] \frac{\partial}{\partial \mathbf{w}} f(\mathbf{x}; \mathbf{w})$$

$$= \left[-y \times \frac{1}{f(\mathbf{x}; \mathbf{w})} + (1 - y) \times \frac{1}{1 - f(\mathbf{x}; \mathbf{w})} \right] f(\mathbf{x}; \mathbf{w})(1 - f(\mathbf{x}; \mathbf{w})) \mathbf{x}$$

$$= \left[-y \times [1 - f(\mathbf{x}; \mathbf{w})] + (1 - y) f(\mathbf{x}; \mathbf{w}) \right] \mathbf{x}$$

$$= \left[-y + f(\mathbf{x}; \mathbf{w}) \right] \mathbf{x}$$

$$= \left[f(\mathbf{x}; \mathbf{w}) - y \right] \mathbf{x}$$

$$\boxed{\frac{\partial}{\partial \mathbf{w}} f(\mathbf{x}; \mathbf{w}) = f(\mathbf{x}; \mathbf{w})(1 - f(\mathbf{x}; \mathbf{w})) \mathbf{x}}$$



$$f(\mathbf{x}) = \frac{\exp(\mathbf{w}^T \mathbf{x})}{\exp(\mathbf{w}^T \mathbf{x}) + 1}$$

$$1 - f(\mathbf{x}) = \frac{1}{\exp(\mathbf{w}^T \mathbf{x}) + 1}$$

$$\begin{aligned}\frac{\partial}{\partial \mathbf{w}} f(\mathbf{x}; \mathbf{w}) &= \frac{\frac{\partial}{\partial \mathbf{w}} [\exp(\mathbf{w}^T \mathbf{x})] (\exp(\mathbf{w}^T \mathbf{x}) + 1) - \exp(\mathbf{w}^T \mathbf{x}) \frac{\partial}{\partial \mathbf{w}} [\exp(\mathbf{w}^T \mathbf{x}) + 1]}{[\exp(\mathbf{w}^T \mathbf{x}) + 1]^2} \\ &= \frac{\exp(\mathbf{w}^T \mathbf{x}) (\exp(\mathbf{w}^T \mathbf{x}) + 1) \frac{\partial}{\partial \mathbf{w}} (\mathbf{w}^T \mathbf{x}) - \exp(\mathbf{w}^T \mathbf{x}) \exp(\mathbf{w}^T \mathbf{x}) \frac{\partial}{\partial \mathbf{w}} (\mathbf{w}^T \mathbf{x})}{[\exp(\mathbf{w}^T \mathbf{x}) + 1]^2} \\ &= \frac{\exp(\mathbf{w}^T \mathbf{x})}{[\exp(\mathbf{w}^T \mathbf{x}) + 1]^2} \mathbf{x} = f(\mathbf{x})(1 - f(\mathbf{x}))\mathbf{x}\end{aligned}$$

$$\boxed{\frac{\partial}{\partial \mathbf{w}} (\mathbf{w}^T \mathbf{x}) = \frac{\partial}{\partial \mathbf{w}} (\mathbf{x}^T \mathbf{w}) = \mathbf{x}}$$

► L2正则函数梯度

- L2正则 : $\Omega(\theta) = \lambda \|\theta\|_2^2 = \lambda \sum_{j=1}^D \theta_j^2$
 - 一阶导数: $\nabla_{\theta} \Omega(\theta) = 2\lambda\theta$
 - 二阶导数: $\nabla_{\theta}^2 \Omega(\theta) = 2\lambda I$

► L1正则函数梯度

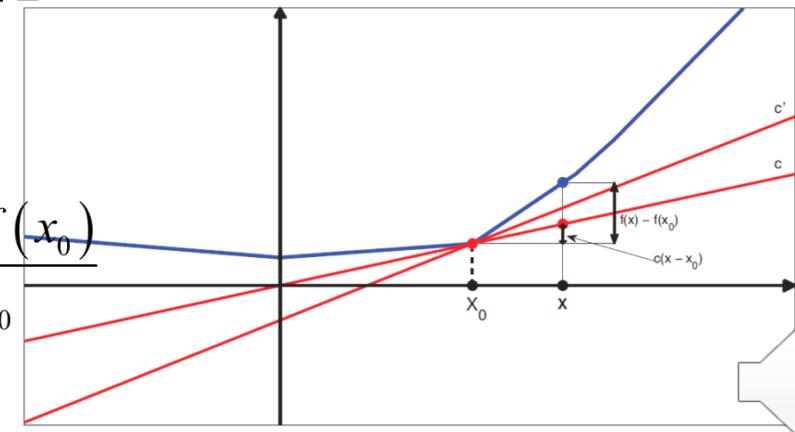
- L1正则: $\Omega(\theta) = \lambda|\theta| = \lambda \sum_{j=1}^D |\theta_j|$
- L1正则在 $\theta_j = 0$ 处不可导，为了处理不平滑函数，扩展导数的表示，定义一个(凸)函数 f 在点 x_0 处的次梯度(subderivative)为一个标量 g ，使得

$$f(x) - f(x_0) \geq g(x - x_0), \forall x \in \mathcal{I}$$

其中 \mathcal{I} 为包含 x_0 的某个区间。

定义区间 $[a, b]$ 的次梯度集合为

$$a = \lim_{x \rightarrow x_0^-} \frac{f(x) - f(x_0)}{x - x_0}, b = \lim_{x \rightarrow x_0^+} \frac{f(x) - f(x_0)}{x - x_0}$$

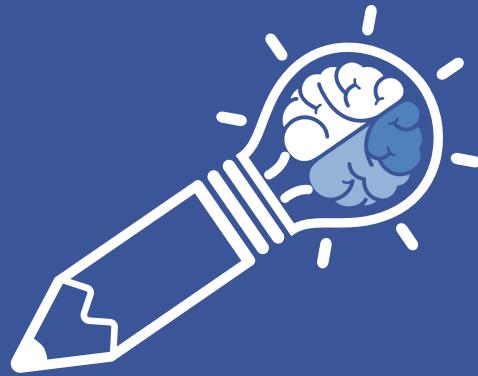


► L1正则函数梯度

- L1正则次梯度为

$$\frac{\partial \Omega(\theta)}{\partial \theta_j} = \begin{cases} \{-1\} & \text{if } \theta_j < 0 \\ [-1, 1] & \text{if } \theta_j = 0 \\ \{1\} & \text{if } \theta_j > 0 \end{cases}$$

- 同标准的微积分类似，可以证明当且仅当 $\theta \in \frac{\partial \Omega(\theta)}{\partial \theta}|_{\hat{\theta}}$ 时， $\hat{\theta}$ 为 Ω 的局部极值点



二、分类回归树



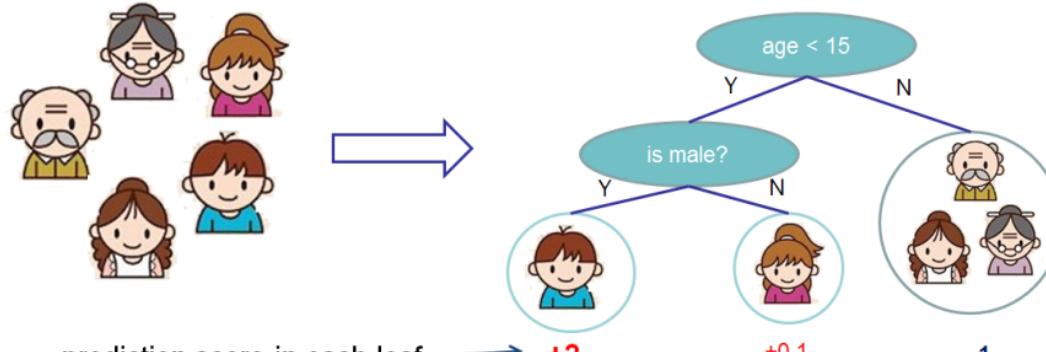
► 分类回归树

- 模型：树（非参数模型）
- 参数：分裂的特征及阈值
- 目标函数
 - 损失函数：L2损失 / GINI指数
 - 正则项：树的节点数目（ L_0 ）、叶子结点分数平方和（ L_2 ）
- 优化
 - 建树
 - 剪枝

► 分类回归树

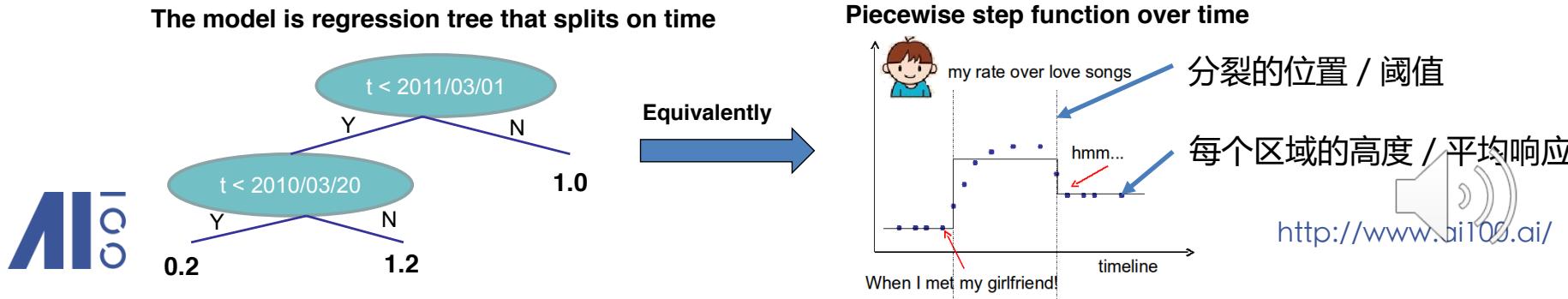
- Classification And Regression Tree (CART) : 机器学习十大算法之一，是一个用于监督学习的**非参数模型**
 - 二分递归分割：将当前样本集合划分为两个子样本集合，使得生成的每个非叶子结点都有两个分支 → 生成的树是**二叉树**

Input: age, gender, occupation, ... Does the person like computer games



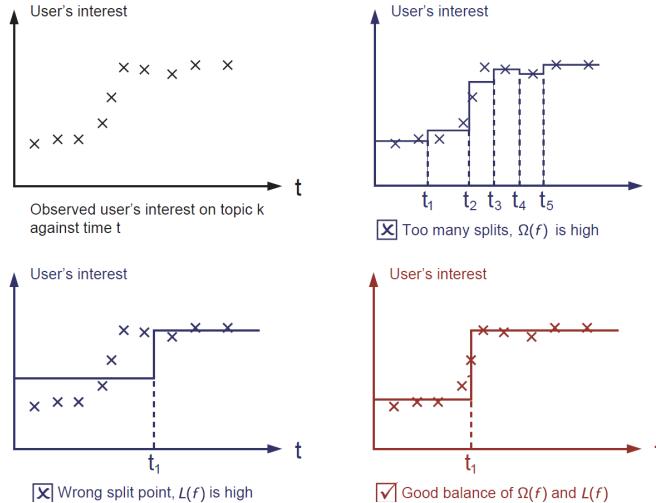
► 例：回归树

- 模型 : $f(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}] = \sum_{m=1}^M w_m \mathbb{I}(\mathbf{x} \in R_m) = \sum_{m=1}^M w_m \phi(\mathbf{x}; \mathbf{v}_m)$
第 m 个区域
的平均响应 第 m 个区域
选择的分裂特征
阈值
根节点到第 m 个
叶子结点的路径
- 参数 : 每次分裂的特征及阈值
- 例 : 预测 t 时刻我是否喜欢Romantic Music



► 例：回归树 (cont.)

- 例：预测 t 时刻我是否喜欢Romantic Music
- 回归问题的目标函数
 - 训练误差 / 损失：函数与样本点的匹配程度
 - 正则：分裂点的数目？每一段的高度？



树模型很容易过拟合
所以很多策略都是在防止模型过拟合，如
提前终止、剪枝、Bagging...



► CART

- 在CART算法中主要分为两个步骤
- (1) 将样本递归划分进行**建树**
 - 损失最小
- (2) 用验证数据进行**剪枝**
 - 正则：减小模型复杂度(节点数目)



► 建树

- 令节点的样本集合为 \mathcal{D} ，对候选分裂 $\boldsymbol{\theta} = (j, t_m)$ ，选择特征 j ，分裂阈值为 t_m ，将样本分裂成左右两个分支 \mathcal{D}_L 和 \mathcal{D}_R
- $\mathcal{D}_L(\boldsymbol{\theta}) = \{(x_i, y_i) | x_{ij} \leq t_m\}$
- $\mathcal{D}_R(\boldsymbol{\theta}) = \{(x_i, y_i) | x_{ij} > t_m\}$
- 分裂原则：分裂后两个分支的样本越纯净越好
 - $G(\mathcal{D}, \boldsymbol{\theta}) = \frac{N_{left}}{N_m} H(\mathcal{D}_L(\boldsymbol{\theta})) + \frac{N_{right}}{N_m} H(\mathcal{D}_R(\boldsymbol{\theta}))$
 - 不纯净性 $G(\mathcal{D}, \boldsymbol{\theta})$ 最小: $\boldsymbol{\theta}^* = argmin_{\boldsymbol{\theta}} G(\mathcal{D}, \boldsymbol{\theta})$
 - 不纯净性度量函数 H 与任务有关

► 建树——回归

- 对回归问题, 集合 \mathcal{D} 的不纯净性为集合中样本的 y 值的差异

- $- H(\mathcal{D}) = \sum_{i \in \mathcal{D}} (\bar{y} - y_i)^2$

- $- \text{其中 } \bar{y} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} y_i$ 为集合中样本的 y 的均值

- $- \text{集合中样本的 } y \text{ 值越接近越纯净}$

- $- \text{相当于损失函数取L2损失, 选择最小L2损失的分裂}$

- $\bullet \text{ L2损失} : L(y - \hat{y}(\boldsymbol{\theta})) = \frac{1}{2} (\hat{y}(\boldsymbol{\theta}) - y)^2 = \frac{1}{2} (\bar{y} - y)^2$

- $\bullet \text{ 预测值为样本均值 } \bar{y} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} y_i \text{ 时L2损失最小}$



► 建树——分类

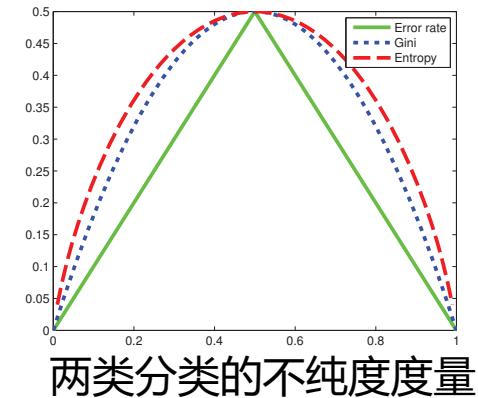
- 分类问题中，分布的估计值取 $\hat{\pi}_c = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{I}(y_i = c)$
 - 即集合中每类样本的比例
- 常用的不纯净度量：

- 错分率： $H(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{I}(y_i \neq \hat{y}) = 1 - \hat{\pi}_{\hat{y}}$

- 熵： $H(\mathcal{D}) = - \sum_{c=1}^C \hat{\pi}_c \log \hat{\pi}_c$

- Gini系数

$$H(\mathcal{D}) = \sum_{c=1}^C \hat{\pi}_c (1 - \hat{\pi}_c) = \sum_c \hat{\pi}_c - \sum_c \hat{\pi}_c^2 = 1 - \sum_c \hat{\pi}_c^2$$



► 建树——停止条件

- 建树过程是一个自顶向下的递归过程
- 递归的停止条件：
 - 分裂带来的损失的减小太小：损失的减少量可定义为

$$\Delta \triangleq \text{cost}(\mathcal{D}) - \left(\frac{|\mathcal{D}_L|}{|\mathcal{D}|} \text{cost}(\mathcal{D}_L) + \frac{|\mathcal{D}_R|}{|\mathcal{D}|} \text{cost}(\mathcal{D}_R) \right)$$

- 树的深度超过了最大深度
- 左 / 右分支的样本分布足够纯净
- 左 / 右分支中样本数目足够少

► 剪枝

- 分类回归树算法容易过拟合，通过剪枝去除部分分支，降低模型复杂度
- 剪枝：给定一个完全树，自底向上进行剪枝，直到根节点
- 剪枝准则：Cost complexity pruning

$$CC(T) = Err(T) + \alpha|T|$$

树的错误率

正则因子 树的节点数目

- 形式同机器学习模型的目标函数： $J(\boldsymbol{\theta}) = \sum_{i=1}^N L(y_i, f(\mathbf{x}_i; \boldsymbol{\theta})) + \Omega(\boldsymbol{\theta})$
- 当 α 从0开始增大，树的一些分支被剪掉，得到不同 α 对应的树
采用交叉验证得到最佳的 α

► 树模型的优点

- 容易解释
- 不要求对特征做预处理
 - 能处理离散值和连续值混合的输入
 - 对特征的单调变换不敏感 (只与数据的排序有关)
 - 能自动进行特征选择
 - 可处理缺失数据
- 可扩展到大数据规模

► 树模型的缺点

- 正确率不高：建树过程过于贪心
 - 可作为Boosting的弱学习器（深度不太深）
- 模型不稳定（方差大）：输入数据小的变化会带来树结构的变化
 - Bagging：随机森林
- 当特征数目相对样本数目太多时，容易过拟合

► Scikit-learn中的Tree

- CART算法的优化实现
 - 建树：穷举搜索所有特征所有可能取值
 - 没有实现剪枝（采用交叉验证选择最佳的树的参数）
 - 分类树：DecisionTreeClassifier
 - 回归树：DecisionTreeRegressor

► DecisionTreeClassifier

- 构造时参数传递参数
- *class sklearn.tree.DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_split=1e-07, class_weight=None, presort=False)*

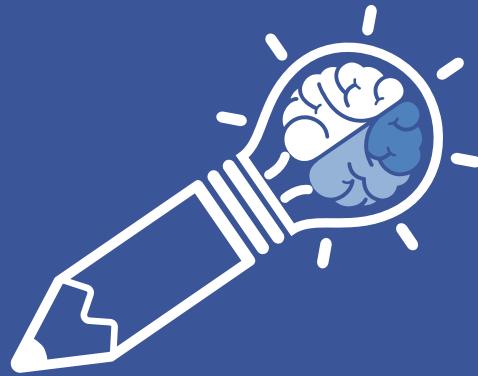
参数

说明

criterion	用来权衡划分的质量。缺省 ‘gini’：即 Gini impurity。或者 ‘entropy’
splitter	划分方式有三种：best, presort-best, random. 缺省：best
max_features	当进行best划分时，考虑的特征数目（个 / 比例）。缺省：None
max_depth	树的最大深度。缺省为：None
min_samples_split	对于一个中间节点（internal node），必须有min个samples才对它进行分割。缺省为：2
min_samples_leaf	每个叶子节点（left node），至少需要有min_samples_leaf个样本。缺省为：1
min_weight_fraction_leaf	叶子节点的样本权重占总权重的比例。缺省为：0
max_leaf_nodes	以最好优先（best-first）的方式使用该值生成树。如果为None：不限制叶子节点的数目。如果不为None，则忽略max_depth。缺省为：None
class_weight	每个类别的权重：{class_label: weight}。如果不给定，所有类别的权重均1. “balanced”模式：自动调整权重。 $n_samples / (n_classes * np.bincount(y))$
random_state	随机种子
presort	是否对数据进行预先排序，以便在fitting时加快最优划分。对于大数据集，使用False，对于小数据集，使用True.

► DecisionTreeRegressor

- `class sklearn.tree.DecisionTreeRegressor(criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_split=1e-07, presort=False)`
- 与分类树的参数基本相同，`criterion`的缺省值为‘mse’，`mean squared error`, 即残差平方和的均值（L2损失）



三、随机森林

► 随机森林 (Random Forest)

- 回归树算法的缺点之一是高方差
- 一种降低算法方差的方式是平均多个模型的预测 : **Bagging**
 - Bootstrap aggregating

► Bagging

- Bootstrap: 对给定有 N 个样本的数据集 \mathcal{D} 进行有放回抽取，得到 N 个样本 \mathcal{D}'
- 对给定有 N 个样本的数据集 \mathcal{D} 进行Bootstrap采样，得到 \mathcal{D}^1 ，在 \mathcal{D}^1 上训练模型 \hat{f}^1
- 上述过程重复 B 次，得到 B 个模型，则 B 个模型的平均为

$$\hat{f}_{avg}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^b(x) \quad \text{aggregating}$$

- 可以证明（略）：Bagging可以降低模型的方差



► 随机森林 (Random Forest)

- 由于只是训练数据有一些不同，对回归树算法进行Bagging得到的多棵树高度相关，因此带来的方差减少有限
- 随机森林通过
 - 随机选择一部分特征
 - 随机选择一部分样本
- 降低树的相关性
- 随机森林在很多应用案例上被证明有效，但牺牲了可解释性
 - 森林：多棵树
 - 随机：对样本和特征进行随机抽取

► 例：Mushroom数据集

- 数据：Kaggle竞赛中直接用22维特征
 - <https://www.kaggle.com/uciml/mushroom-classification>
- 运行2_Mushroom_CART.ipynb
 - 学习Logistic 回归、CART、随机森林模型的使用
 - 交叉验证选择模型（超参数）

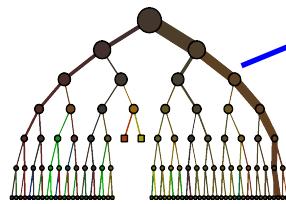
A brief history of forests



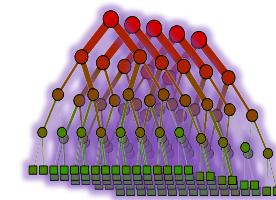
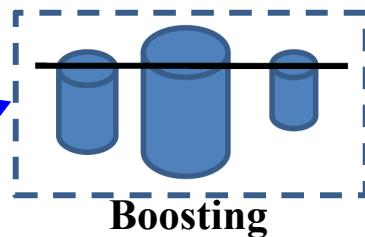
**Jerome H.
Friedman**



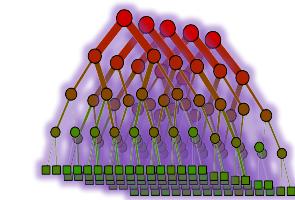
Leo Breiman



CART



GBDT



Random Forest

1984

1996 1998

1999

►下节课预告

- Boosting
- Gradient Boosting
- XGBoost



THANK YOU

北京智百科技有限公司



<http://www.ai100.ai/>

► 偏导数

- 一个多变量的函数的偏导数：是函数关于其中一个变量的导数，而保持其他变量恒定
 - 相对于全导数，在其中所有变量都允许变化
- 全导数：对函数 $f(\mathbf{x}; \boldsymbol{\theta})$ ，全导数为 $\frac{df}{d\boldsymbol{\theta}}$
- 偏导数：对函数 $f(\mathbf{x}; \boldsymbol{\theta})$ ，对变量 θ_j 的偏导数为
- $$\frac{\partial f(\mathbf{x}; \boldsymbol{\theta})}{\partial \theta_j} = \lim_{t \rightarrow 0} \frac{f(\mathbf{x}; \theta_1, \dots, \theta_j + t, \dots, \theta_D) - f(\mathbf{x}; \theta_1, \dots, \theta_j, \dots, \theta_D)}{t}$$

► 方向导数

- 方向导数表示函数在某点沿着某个向量方向上的的导数，描绘了函数在该点附近沿着该方向变动时的瞬时变化。方向导数是偏导数的推广。
- 函数在方向 \mathbf{a} 上的方向导数定义为
- $$\frac{\partial f(\mathbf{x}; \boldsymbol{\theta})}{\partial \theta_j} = \lim_{t \rightarrow 0} \frac{f(\mathbf{x}; \boldsymbol{\theta} + t\mathbf{a}) - f(\mathbf{x}; \boldsymbol{\theta})}{t}$$

► 梯度

- 梯度表示函数在某个点的位置法向量，表示函数下降最快或者上升最快的方向。
- 在单变量的实值函数的情况，梯度只是导数
 - 对线性函数，也是线的斜率
- 对多变量函数，梯度定义为：

$$\bullet \nabla_{\theta} f(\mathbf{x}; \boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \theta_1} \\ \dots \\ \frac{\partial f}{\partial \theta_D} \end{bmatrix} = \frac{\partial f(\mathbf{x}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$